A

Please type a plus sign (+) inside this box [+]

PTO/SB/05 (12/97)
Approved for use through 09/30/00. OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 CFR 1.53(b)

Attorney Docket No. 042390.P7389 Total Pages 3

First Named Inventor or Application Identifier Bradford H. Needham

Express Mail Label No. EL143564687US

ADDRESS TO: **Assistant Commissioner for Patents**
Box Patent Application
Washington, D. C. 20231

APPLICATION ELEMENTS
See MPEP chapter 600 concerning utility patent application contents.

1. X Fee Transmittal Form
(Submit an original, and a duplicate for fee processing)

2. X Specification (Total Pages 27)
(preferred arrangement set forth below)
- Descriptive Title of the Invention
- Cross References to Related Applications
- Statement Regarding Fed sponsored R & D
- Reference to Microfiche Appendix
- Background of the Invention
- Brief Summary of the Invention
- Brief Description of the Drawings (if filed)
- Detailed Description
- Claims
- Abstract of the Disclosure

3. X Drawings(s) (35 USC 113) (Total Sheets 3)

4. X Oath or Declaration (Total Pages 4)

a. X Newly Executed (Original or Copy)

b. ___ Copy from a Prior Application (37 CFR 1.63(d))
(for Continuation/Divisional with Box 17 completed) **(Note Box 5 below)**

i. ___ DELETIONS OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b).

5. ____ Incorporation By Reference (useable if Box 4b is checked)
The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.

6. ____ Microfiche Computer Program (Appendix)

7. _____ Nucleotide and/or Amino Acid Sequence Submission
(if applicable, all necessary)
a. ____ Computer Readable Copy
b. ____ Paper Copy (identical to computer copy)
c. ____ Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

8. __X__ Assignment Papers (cover sheet & documents(s))

9. _____ a. 37 CFR 3.73(b) Statement (where there is an assignee)
_____ b. Power of Attorney

10. _____ English Translation Document (if applicable)

11. _____ a. Information Disclosure Statement (IDS)/PTO-1449
_____ b. Copies of IDS Citations

12. _____ Preliminary Amendment

13. __X__ Return Receipt Postcard (MPEP 503) (Should be specifically itemized)

14. _____ a. Small Entity Statement(s)
_____ b. Statement filed in prior application, Status still proper and desired

15. _____ Certified Copy of Priority Document(s) (if foreign priority is claimed)

16. _____ Other: _____

17. **If a CONTINUING APPLICATION,** check appropriate box and supply the requisite information:
___ Continuation ____ Divisional ____ Continuation-in-part (CIP)
of prior application No: _____

18. **Correspondence Address**

_____ Customer Number or Bar Code Label _____
(Insert Customer No. or Attach Bar Code Label here)

or

__X__ Correspondence Address Below

NAME Bradley J. Bereznak
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

ADDRESS 12400 Wilshire Boulevard
Seventh Floor

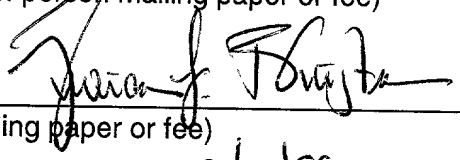
CITY Los Angeles STATE California ZIP CODE 90025-1026

Country U.S.A. TELEPHONE (408) 720-8598 FAX (408) 720-939

"Express Mail" mailing label number: EL143564687US
Date of Deposit: September 21, 1999

I hereby certify that I am causing the foregoing transmittal, copy thereof, attached fee transmittal, copy thereof, checks in the amounts of $838.00 and $40.00, attached specification, drawings, declaration, and assignment, to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that these papers and fee have been addressed to the Assistant Commissioner for Patents, Box Patent Application, Washington, D. C. 20231

Vivian Y. Buijten
(Typed or printed name of person mailing paper or fee)



(Signature of person mailing paper or fee)

9/21/99

(Date signed)

UNITED STATES PATENT APPLICATION

for

A MOTION DETECTING WEB CAMERA SYSTEM

Inventor:

Bradford H. Needham

Prepared by:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, LLP
12400 Wilshire Boulevard
Los Angeles, CA 90025-1026
(408) 720-8598

Attorney Docket No: 42390.P7389

A MOTION DETECTING WEB CAMERA SYSTEM

RELATED APPLICATION

This application is related to Application No. _____ entitled, "A Web Microphone System", filed concurrently herewith, and Application No. 09/124,179 entitled, "Digital Opaque Projector for Easy Creation and Capture of Presentation Material", filed July 28, 1998, which applications are assigned to the assignee of the present application.

FIELD OF THE INVENTION

The present invention relates generally to the field of video capture; more particularly, to video camera systems having information processing capabilities for uploading pictures to a web server.

BACKGROUND OF THE INVENTION

A web camera (i.e., "webcam") system consists of a video camera plus software that runs on a personal computer to periodically upload an image from the camera to a web page. The basic purpose of a web camera system is to post a reasonably live picture on a user-specified web page. Many webcam systems upload images on a periodic basis; for example, uploading an image once per hour.

Figure 1 illustrates a web camera system 10, which is typical of the prior art. System 10 includes a video camera 11 that outputs a captured video image to a personal computer (PC) 12. Software running on PC 12 functions to periodically upload the captured video image to an Internet web page (i.e., a web server) shown in Figure 1 by block 13. Internet service providers (ISPs) commonly provide their patrons with a certain allocation of web page space for

personal use. This allows the user to upload images onto their web page periodically; with the frequency of uploading being dependent on the particular type of connection offered by the ISP.

Presently, there are two shareware products in existence that relate to web cameras: Ispy ™ and Webcam32™. The Ispy webcam software functions to grab video images, save them as JPEG files, and then send the saved images automatically to a user-specified home page via the connection provided by the users' ISP. Ispy runs under Windows™ 95, Windows 98 and Windows NT 4.0; it also works with any video for Windows-compatible cameras and frame grabbers. Webcam32 is a Windows 95, Windows 98 and Windows NT application that allows video camera images to be displayed within a web page. Like Ispy, Webcam32 software is able to upload images to a web server to allow images to be obtained directly from the page. Both of these products include various simple image-processing features such as captioning of photos, day/time stamping, and text additions.

Webcam32™ software also offers rudimentary motion detection, which is of primary use in security surveillance applications. For example, the Webcam32™ software allows images to be uploaded when, say, 25% of the pixels in the image frame change from one image frame to the next. Although this motion-detecting feature of the software product is useful in limited types of motion detection applications (e.g., security surveillance), it is not useful for different applications. For example, if the web camera system is intended for use in observing and recording wildlife activity, then this type of rudimentary motion detection does not work well.

Another problem with today's webcam systems is the conflict between the desire to minimize the number of times the web camera contacts the ISP and the

need to capture "interesting" pictures (i.e., those containing certain kinds of motion). Most security surveillance type of web camera systems have a low threshold that results in the taking of many pictures whenever activity is detected. Uploading many pictures onto a web page presents a serious bandwidth problem.

Furthermore, existing products such as Ispy and Webcam32 only provide the ability to capture images on a given schedule, e.g., once per hour, or whenever motion occurs, regardless of how often. If set to capture on a predetermined schedule, images that may be of interest to the user may end up being ignored. On the other hand, if the software is set to upload a video image whenever motion is detected, scenes containing frequent motion can tie up the user's phone lines.

Thus, there is a need for a web camera system that overcomes the problems inherent in the prior art.

SUMMARY OF THE INVENTION

The present invention is a camera system for connection to a web server. The system comprises a video camera and a processor that periodically uploads images captured by the video camera in accordance with one of a plurality of motion detection algorithms. A first motion detection algorithm captures a current image frame when a pixel comparison between successive image frames exceeds a predetermined threshold.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be understood more fully from the detailed description which follows and from the accompanying drawings, which however, should not be taken to limit the invention to the specific embodiments shown, but are for explanation and understanding only.

Figure 1 is a diagram of a prior art video capture system.

Figure 2 is an example of an application of the present invention.

Figure 3 is a conceptual diagram of various motion-detecting algorithms utilized in accordance with one embodiment of the present invention.

Figure 4 is a flowchart illustrating one of the motion-detecting algorithms utilized in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION

A web camera system that operates in accordance with one of a plurality of motion detection algorithms is described. In the following description, numerous specific details are set forth, such as specific operating modes, procedures, circuit elements, etc., in order to provide a thorough understanding of the invention. It will be obvious, however, to one skilled in the art, that these specific details may not be needed to practice the present invention.

The state-of-the-art of web camera systems is such that there exists a conflict between the desire to minimize the number of times the web camera system dials up the Internet service provider and the desire to capture "interesting" pictures (e.g., those containing certain particular kinds of motion). As previously discussed, existing software products utilized to capture video images permit some rudimentary motion detection. These programs are utilized in applications concerned primarily with uploading images the instant motion is detected. While such programs are suitable for use in applications such as security surveillance systems, they suffer problems when used in different applications, e.g., observation of wildlife activity.

Figure 2 illustrates three image frames 21-23 that may be captured utilizing a web camera system. In this case, the picture of interest is a bird feeding at a bird feeder station. The web camera is directed at the station, and operates to upload captured images either periodically, or in response to detected motion, or both. The problem that exists with prior art web camera systems that periodically upload images, say, every half-hour, is that a bird may arrive and leave many times within that time interval. In the event that a bird is not present at the scheduled image capture time, the only picture that will be

captured is one of an empty bird feeder. By way of example, this is the situation represented by image frame 21.

If a motion detecting feature is included in the web camera system, the taking of a picture may be triggered each time movement above a certain threshold is detected. In this case, when the number of pixels between two successive pictures changes (above a predetermined threshold) a new picture is captured and uploaded to the user's specified web site. By way of example, frame 22 of Figure 2 illustrates the arrival of a bird at the bird feeder, which triggers the video capture of an image frame. Note that many web camera systems in use today typically have low threshold settings that result in the taking of many pictures when the slightest activity is detected. This results in a bandwidth problem for the connection to the Internet service provider.

Another concern relates to movement of the bird when it takes flight to leave the bird feeder station. Existing web camera systems with motion detecting features will trigger on this type of motion. Unfortunately, the last video image captured as a result of this type of motion is an empty bird feeder station, as represented by frame 23. In other words, the most recently captured image for uploading to the user's web site is that of the empty bird feeder, rather than the desired image of wildlife activity.

The present invention solves the problem of motion detection and timed update by uploading one image each predetermined interval -- selecting the best candidate video image that occurred during any given interval. The camera system includes a video camera coupled to a processor that operates in accordance with one of a plurality of motion detection algorithms to select an image for uploading to the user's web site.

The basic algorithm is as follows. Motion is first detected by performing a pixel comparison between successive image frames. When the pixel comparison between a current image frame and a previous image frame exceeds a predetermined threshold level, software running on the processor saves the current picture as a candidate for uploading. A simple frame grabber technology may be utilized for capture of the video image.

The processor may be programmed to periodically upload the current or last candidate image. For example, a typical web camera system may operate by uploading the last candidate image once every hour. If no motion is detected in the past hour, there are two options: either no uploading of any image, or upload a current image captured by the camera regardless of whether motion is detected.

Figure 3 illustrates a conceptual diagram of various detecting algorithms utilized in accordance with one embodiment of the present invention. Details of this embodiment are provided below and also in the attached Appendix, which contains a code listing of a software program that implements these algorithms.

With reference to Figure 3, a camera outputs a color image, which is then converted to a black-and-white (B/W) pixel representation of a current image frame. Three different modes of motion detection are then provided to capture specific types of events. In the basic mode of motion detection, the web camera system selects the most recent image containing motion above a certain threshold. This image represents the candidate image for uploading to the user-specified web server. As previously discussed, the basic mode of motion detection involves a straightforward pixel-by-pixel comparison between a current frame and a previously captured image frame.

A second type of motion detection mode of operation is referred to as "stable-change" detection. The stable-change mode of motion detection is designed to capture persistent changes in a scene, while ignoring relatively simple motions. For example, the stable-change mode of operation is useful in video conferencing applications where the video camera is aimed at a desk or whiteboard where a speaker is placing written subject matter or other objects in front of the camera to facilitate discussion. In such applications, it is useful to capture the notes written on a board, or otherwise presented in front of the camera, while ignoring extraneous motion such as writing on the board, finger pointing, etc.

The stable-change mode of operation is aimed at detecting stable changes to a particular scene and operates in accordance with an algorithm that captures an image frame a certain time period following a last detected motion event. The time duration may be programmable, timed, or fixed depending on what is being viewed and what is to be captured. In the video conferencing application discussed above, software running on the processor operates to capture a video image frame and copy or upload it to remote sites whenever a new writing (or other object for presentation) is placed in front of the video camera. The stable-change mode of operation ignores constant ongoing activity in the field of view.

As can be seen in Figure 3, the stable-change algorithm compares a current frame against a last stable frame and selects as a candidate picture the last stable frame when no motion has been detected for a certain duration (e.g., two seconds).

The third mode of motion detection operation is referred to in this application as "novel" motion detection. The novel motion detection mode of

operation solves the problem that arises in certain applications such as observation of wildlife activity wherein the motion of a bird arriving at a feeder is very similar to the motion of the bird departing from the feeder. A webcam system operating in accordance with only a basic motion detection algorithm -- which simply saves the most recent image with motion -- cannot distinguish between these two types of events. In other words, the basic motion detection algorithm captures the image of the just vacated bird feeder for uploading to the web server, rather than the picture of the bird that left, simply because it triggers on the last motion detected.

In solving this problem, the novel motion detection algorithm compares an image that contains motion against the most recent stable image, as described above. Images that are not substantially different from the stable image frame are ignored.

Figure 3 illustrates that novel motion detection involves not only detection of motion, based on a pixel comparison of a current frame against a previous frame, but also a comparison between the current frame and a last stable frame. For example, as applied to the empty bird feeder problem, the last stable frame is that of the recently vacated bird feeder. According to this third algorithm, motion is recorded instead of the absence of motion. That is, the detection of motion, based on a pixel comparison between the current frame and previous frame, triggers capture of a current image frame in a buffer. The buffer may be any one of a variety of buffers, such as a circular buffer, with a capacity to store a sufficient number of image frames.

In this mode, each time motion is detected, the current image is captured into a circular buffer. In other words, in the novel motion detection mode of operation, pictures are captured into a buffer at times other than the last stable

frame. The detection of a last stable frame triggers the uploading of an image from the circular buffer that was captured some predetermined time prior to the triggering event. In the bird feeder example, the uploaded image might be an image frame captured several seconds prior to the last detected motion, e.g., an image of the bird prior to leaving.

In one embodiment of the present invention the selected image represents a candidate picture that may be uploaded to a web server at a predetermined interval. Generally speaking, the user may set the interval for uploading the last candidate picture, as well as the particular mode of motion detection to be utilized.

Figure 4 is a flow chart illustrating the novel motion detection algorithm in accordance with one embodiment of the present invention. The flow chart begins at step 31, which indicates the capture of a current image frame by the video camera. If a colored video camera is utilized, the color image may be transformed to a black and white pixel representation. At step 33, a pixel comparison is made between the current frame and a previously captured frame. If the pixel comparison indicates that the number of pixels between the two frames exceeds a predetermined threshold, then the algorithm proceeds to step 34. If the predetermined threshold is not exceeded (i.e., no motion is detected) the flow chart returns to step 31.

When basic motion is detected, the pixel comparison causes a motion signal to be asserted by logic circuitry in the processor. This is illustrated in Figure 4 by step 34. Assertion of the motion signal causes the current image to be loaded into a candidate buffer which holds the most recent images for periodic uploading to a web site.

Both the current image frame and the previous image frame may be held in separate buffers after being captured by the video camera. In this particular mode of operation, a circular buffer may be utilized as a candidate buffer for holding the most recent images captured responsive to the motion signal. The storing of the image frames in the circular buffer is represented in Figure 4 by step 35.

Step 36 is a determination of whether a stable image frame is detected. If not, the algorithm returns to the beginning step 31. On the other hand, if a stable frame is detected, then one of the stored frames is selected for uploading to a web server. The particular frame that may be chosen is the frame that occurred a predetermined time prior to the detection of a stable frame. For example, in the bird feeder example, it is useful to select an image that was captured several seconds prior to the last detected motion of the bird leaving the feeder.

CLAIMS

I claim:

1. A camera system for connection to a web server comprising:

a video camera;

a processor that periodically uploads images captured by the video camera in accordance with one of a plurality of motion detection algorithms, a first motion detection algorithm capturing a current image frame when a pixel comparison between successive image frames exceeds a predetermined threshold.

2. The camera system of claim 1 wherein the processor uploads the current image frame at programmed intervals.

3. The camera system of claim 1 wherein the plurality of motion detection algorithms further comprises a second motion detection algorithm that captures a stable frame after a certain duration has elapsed since the predetermined threshold has been exceeded.

4. The camera system of claim 1 wherein the plurality of motion detection algorithms further comprises a third motion detection algorithm that captures a recent motion frame that occurs a predetermined time period prior to the occurrence of a stable frame, the stable frame occurring after a certain duration has elapsed since the predetermined threshold has been exceeded.

5. The camera system of claim 3 wherein the plurality of motion detection algorithms further comprises a third motion detection algorithm that captures a

recent motion frame that occurs a predetermined time period prior to the occurrence of the stable frame.

6. The camera system of claim 4 wherein the processor includes a circular buffer to successively store motion captured in image frames in which the predetermined threshold is exceeded.

7. A web camera system for uploading pictures to a web site comprising:

a video camera;

a processor coupled to the video camera, the processor including:

a current frame buffer to hold a current image captured by the video camera;

a previous frame buffer to hold a previous image captured prior to the current image;

a candidate buffer to hold a most recent image for periodic uploading to the web site;

logic circuitry to perform a pixel comparison between the current image and the previous image, the logic circuitry asserting a motion signal when the pixel comparison exceeds a predetermined threshold;

the processor operating according to one of a plurality of modes, in a first mode of operation the current image is loaded into the candidate buffer responsive to the motion signal.

8. The web camera system of claim 7 wherein in a second mode of operation the candidate buffer is loaded with the current image after a certain duration has elapsed following assertion of the motion signal.

9. The web camera system of claim 8 further comprising:

a circular buffer to store successive current images when the motion signal is asserted;

in a third mode of operation the processor selecting one of the current images stored in the circular buffer for loading into the candidate buffer once the motion signal has been de-asserted for a predetermined time.

10. The web camera system of claim 7 further comprising:

a circular buffer to store successive current images when the motion signal is asserted;

in a third mode of operation the processor selecting one of the current images stored in the circular buffer for loading into the candidate buffer once the motion signal has been de-asserted for a predetermined time.

11. A method of operating a web camera system comprising:

capturing a current image frame from a video camera;

asserting a motion detection signal when a pixel comparison between the current image and a previous image frame exceeds a predetermined threshold;

storing in a buffer successive image frames captured from the video camera while the motion detection signal is asserted;

de-asserting the motion detection signal when the predetermined threshold is no longer exceeded for the current image frame;

selecting from the buffer a certain one of the successive image frames as a candidate picture once the motion detection signal has been de-asserted for a certain duration; and

uploading the candidate picture to a web site.

12. The method according to claim 11 wherein the buffer is a circular buffer having a capacity to store a plurality of image frames.

13. The method according to claim 11 wherein the uploading step is performed at periodic time intervals.

14. The method according to claim 11 wherein the certain one of the successive image frames is stored a predetermined time before a last image frame is stored in the buffer prior to de-assertion of the motion detection signal.

15. A method of operating a web camera system comprising:

capturing a current image frame from a video camera;

asserting a motion detection signal when a pixel comparison between the current image and a previous image frame exceeds a predetermined threshold;

storing in a buffer successive image frames captured from the video camera while the motion detection signal is asserted;

de-asserting the motion detection signal when the predetermined threshold is no longer exceeded for the current image frame;

selecting as a candidate picture either:

(i) the current image when the motion detection signal is asserted;

(i) the current image a first duration following de-assertion of the motion detection signal; or

(ii) a certain one of the successive image frames from the buffer once the motion detection signal has been de-asserted for a second duration; and

uploading the candidate picture to a web site.

16. The method according to claim 11 wherein the buffer is a circular buffer having a capacity to store a plurality of image frames.

17. The method according to claim 11 wherein the uploading step is performed at periodic time intervals.

18. The method according to claim 11 wherein the certain one of the successive image frames is stored a predetermined time before a last image frame is stored in the buffer prior to de-assertion of the motion detection signal.

ABSTRACT OF THE DISCLOSURE

A camera system that includes a video camera and a processor, which periodically uploads images captured by the video camera to a web server in accordance with one of a plurality of motion detection algorithms. A first motion detection algorithm captures a current image frame when a pixel comparison between successive image frames exceeds a predetermined threshold.

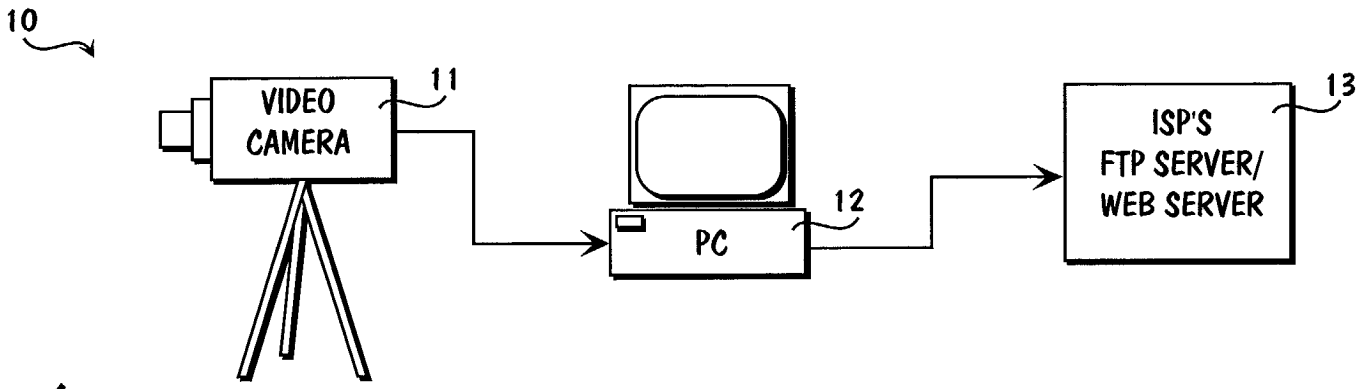


FIG. 1 (PRIOR ART)

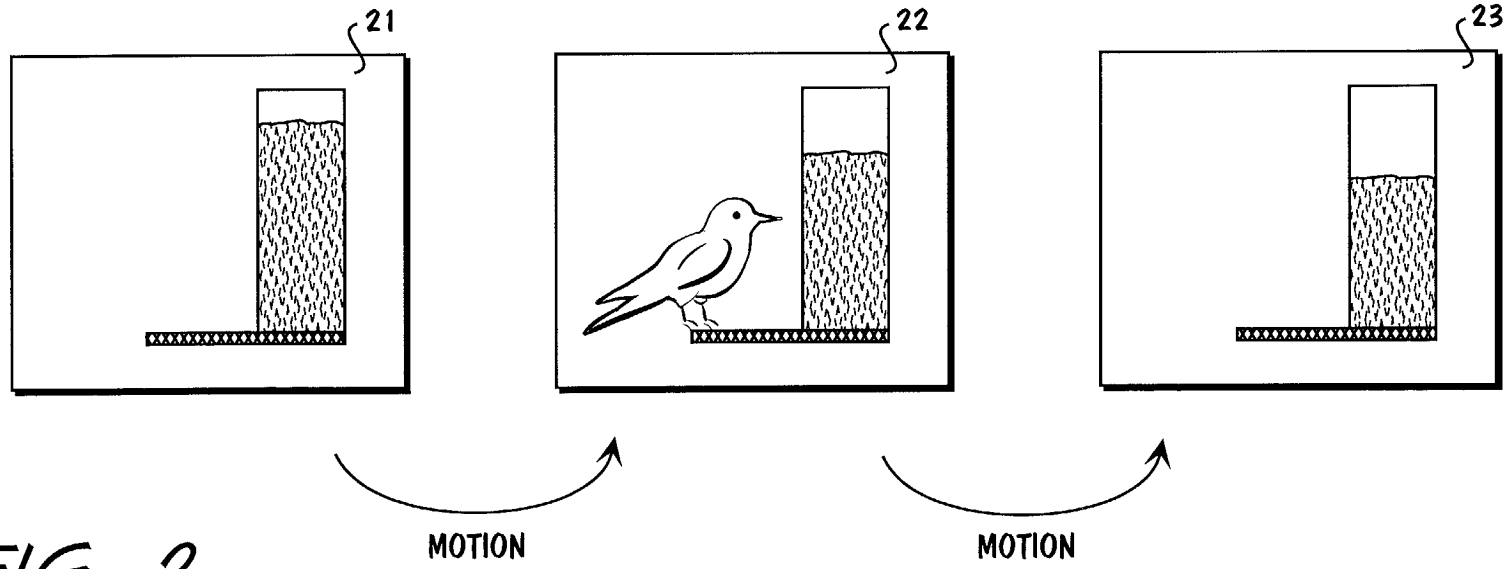


FIG. 2

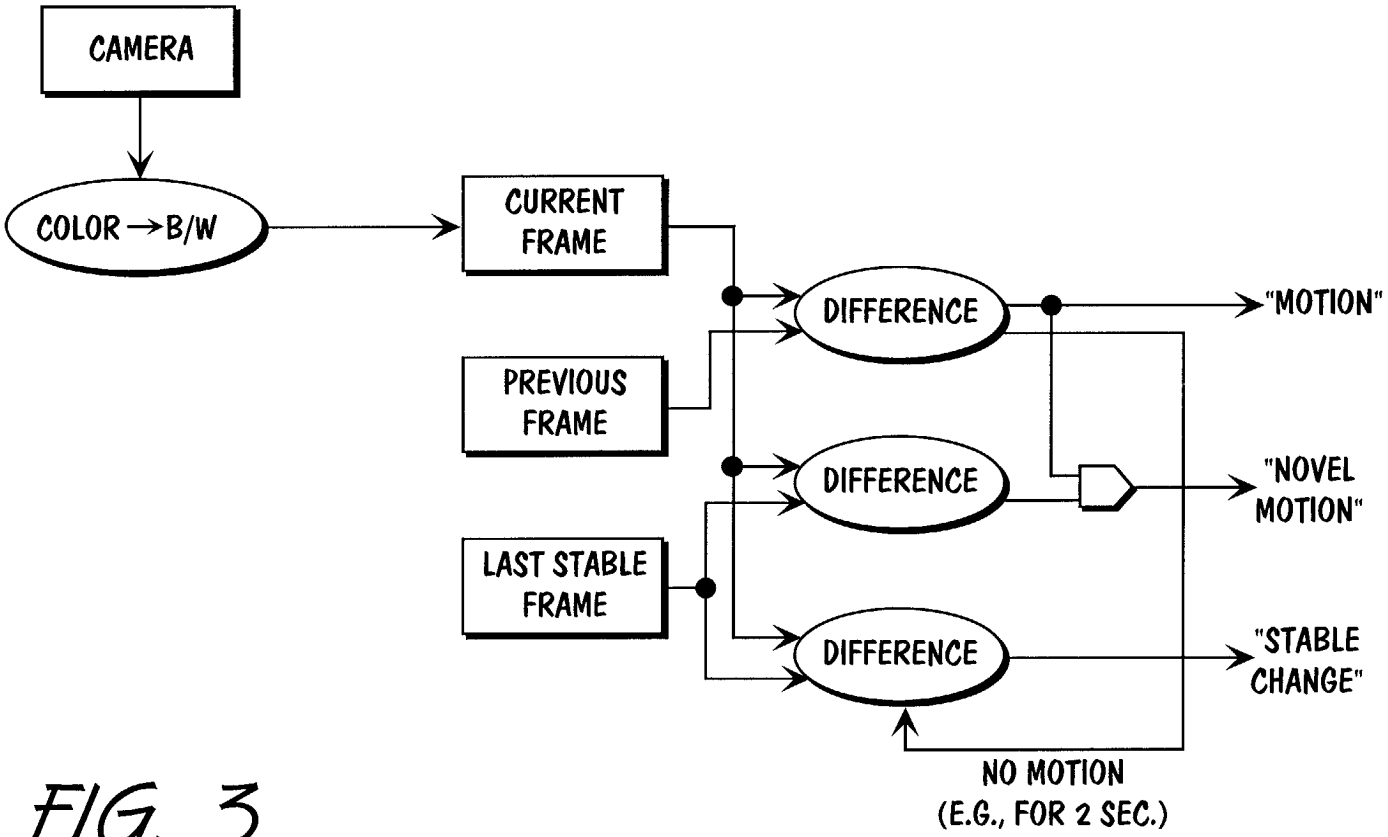

CAMERA
COLOR → B/W
CURRENT FRAME
PREVIOUS FRAME
LAST STABLE FRAME
DIFFERENCE
DIFFERENCE
DIFFERENCE
"MOTION"
"NOVEL MOTION"
"STABLE CHANGE"
NO MOTION
(E.G., FOR 2 SEC.)

FIG. 3

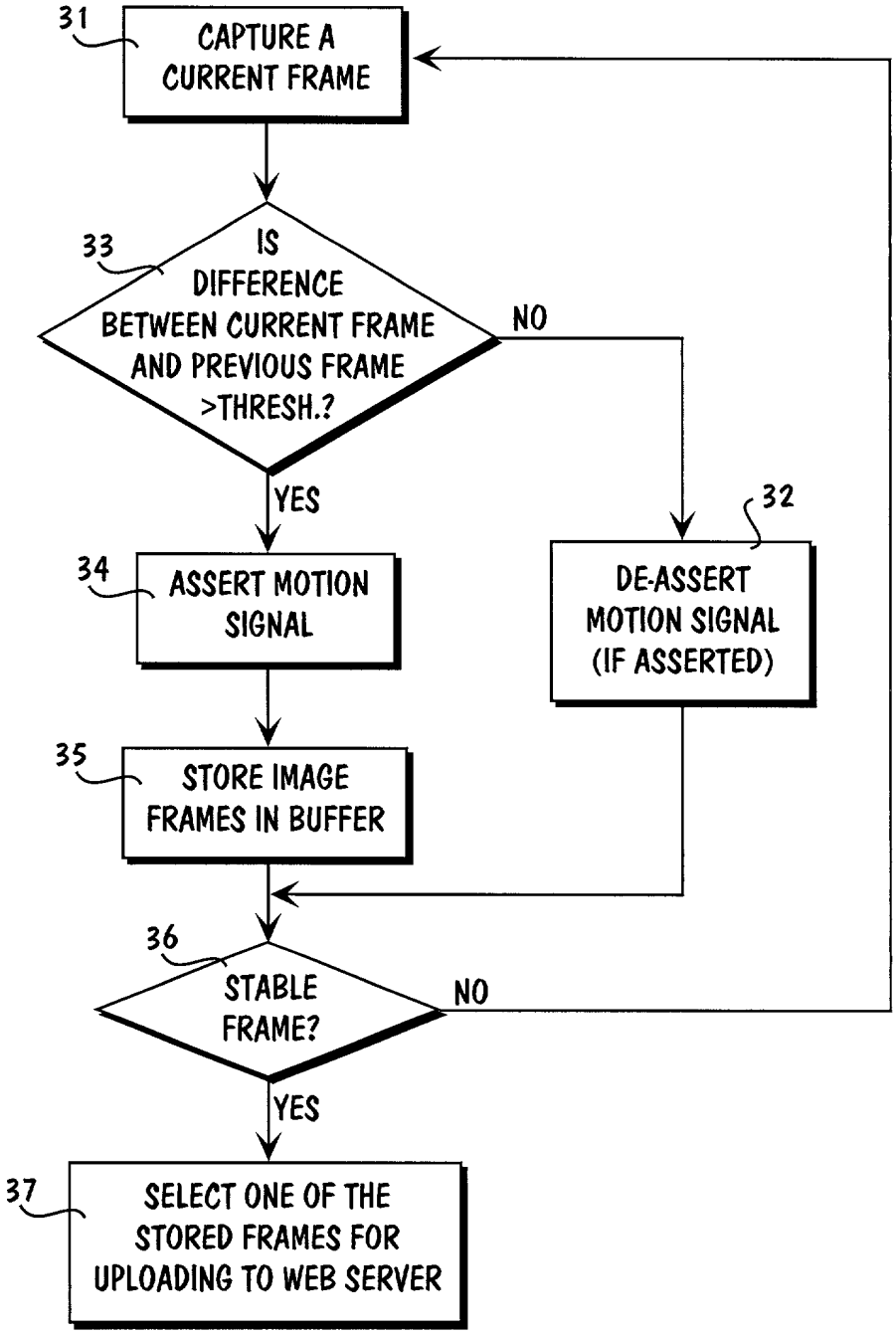

31
CAPTURE A CURRENT FRAME
33
IS DIFFERENCE BETWEEN CURRENT FRAME AND PREVIOUS FRAME >THRESH.?
NO
YES
32
DE-ASSERT MOTION SIGNAL (IF ASSERTED)
34
ASSERT MOTION SIGNAL
35
STORE IMAGE FRAMES IN BUFFER
36
STABLE FRAME?
NO
YES
37
SELECT ONE OF THE STORED FRAMES FOR UPLOADING TO WEB SERVER

FIG. 4

```
// Called when an ID_CONTROL_NEW_FRAME message is received
// as a result of the frame callback getting a new frame.
void CSnapshotCtrl::OnControlNewframe()
{
    DWORD startTime;        // time (ms since bootup) when we started processing this frame.
    DWORD endTime;
    DWORD msTimeSincePrev;  // time (ms) elapsed since the previous frame was snapped.
    BOOL sendFrame;         // "send this frame as a snapshot"
    BOOL motionOccurred;    // "motion occurred between the previous frame and the current frame."
    BOOL noveltyOccurred;   // "motion occurred between the stable frame and the current frame."
    BOOL changeOccurred;    // "a stable change happened"
    DWORD msIdle;           // time (ms) elapsed since movement was detected.
    int diffThreshold;      // the absolute threshold to use in the difference calculation.
    CRect moveRect;         // the motion rectangle, in video coordinates.
    double pcNumChanged;    // number of pixels that have changed (vs. moved), in percent.
    CTime ctNow;            // the time the frame was taken, in seconds since the epoch.
    CString cStrTag;        // text label for the current picture.
    double fDiff;
    IPLStatus iplStat;
    IplROI *pRoi = NULL;    // region-of-interest for differences of the images (or NULL if none).
    IplROI *prevRoi1, *prevRoi2, *prevRoi3; // previous region-of-interest for images we mess with

    LPCSTR pMsg;            // points to a static error message.
    IplLUT *pLut[3];        // pointers to each lut[] entry.
    IplLUT lut[3];          // the Lookup-table for each channel.
    int lutKey[256 * 3];    // the key values for each channel's LUT.
    int lutValue[256 * 3];  // the value values for each channel's LUT.
    int lutIdx;
    int keyIdx;

    sendFrame = FALSE;
    motionOccurred = FALSE;
    noveltyOccurred = FALSE;
    changeOccurred = FALSE;
    startTime = timeGetTime();
    ctNow = CTime::GetCurrentTime();

    // Keep our callback from overwriting our data.

    m_frameBusy = TRUE;

    // If the video was closed a while ago, ignore this message.
    // If we've already processed this frame, ignore it.
    // This helps when we are capturing frames faster than we can process them.

    if (!m_pFrameCam || !m_pIplIn || !m_readyFrameIn) {
        m_frameBusy = FALSE;
        return;
    }

    // If the camera format is compressed, decompress it first so that IIPL can handle it.
    // Otherwise, just copy it.

    if (m_hicD) {
        if (ICERR_OK != ICDecompress(m_hicD, 0L,
          &m_pFmtCam->bmiHeader, m_pFrameCam,
          &m_pFmtIn->bmiHeader, m_pFrameIn)) {
            Stop();
            AfxMessageBox("Frame decompress failed.");
            m_readyFrameIn = FALSE;
            m_frameBusy = FALSE;
```

```
            return;
        }
    } else {
        if (m_pFmtCam->bmiHeader.biSizeImage != m_pFmtIn->bmiHeader.biSizeImage) {
            TRACE("OnControlNewFrame: camera imagesize != input imagesize\n");
        }
        memcpy(m_pFrameIn, m_pFrameCam, m_pFmtIn->bmiHeader.biSizeImage);
    }

    // Convert the image to Intel Image Processing Library format for further processing.

    iplStat = iplConvertFromDIBSep(&m_pFmtIn->bmiHeader, (const char *) m_pFrameIn, m_pIplIn);
    if (iplStat != 0) {
        Stop();
        AfxMessageBox("Couldn't convert camera output to RGB-24 format");
        m_readyFrameIn = FALSE;
        m_frameBusy = FALSE;
        return;
    }

    // If the user wants rotate the image, do that.
    // (mirroring, on the other hand, is a preview function performed in OnDraw).
    // A 180-degree rotation is equivalent to mirroring horizontally and vertically.

    if (m_rotate) {
        iplMirror(m_pIplIn, m_pIplIn, -1);
    }

    // Figure out how much time has elapsed since the previous snapshot,

    msTimeSincePrev = startTime - m_msPrevSnapTime;

    // Calculate our absolute pixel difference threshold:
    // It's the per-pixel noise * 2 (because noise is additive) * 255 (because that's our luminanc
e range)
    // divided by 100 because our noise is expressed in percent,
    // rounded to the nearest integer.

    diffThreshold = (int) ((m_fPixelNoise * 2.0 * 255.0) / 100.0 + 0.5);

    // Find our motion-detection rectangle in valid video coordinates.

    moveRect = m_rDetect;
    if (moveRect.left < 0) {
        moveRect.left = 0;
    }
    if (moveRect.top < 0) {
        moveRect.top = 0;
    }
    if (moveRect.right > m_pIplIn->width) {
        moveRect.right = m_pIplIn->width;
    }
    if (moveRect.bottom > m_pIplIn->height) {
        moveRect.bottom = m_pIplIn->height;
    }

    switch (m_snapMode) {
    case SNAP_BULB:
        // bulb mode is really a "don't take a picture unless forced" -- do nothing.
        break;
    case SNAP_TIME_LAPSE:
        // Time lapse is really "try to send every frame, but all modes are throttled by m_msTimeL
```

```
apse"
        sendFrame = TRUE;
        break;
    case SNAP_ON_MOTION:
    case SNAP_ON_NOVELTY:
    case SNAP_ON_CHANGE:
        // motion- and change-detection modes are handled below.
        break;
    default:    // unknown snap mode.
        Stop();
        ASSERT(FALSE);
    }

    // If we're going to do any sort of motion/change-detection on this frame,
    // do it.

    switch (m_snapMode) {
    case SNAP_BULB:
    case SNAP_TIME_LAPSE:
        // these modes aren't motion modes.
        break;
    case SNAP_ON_MOTION:
    case SNAP_ON_NOVELTY:
    case SNAP_ON_CHANGE:

        // The motion-detection modes require limiting the detection to the motion rectangle.
        // Create a region-of-interest for the motion rectangle.

        pRoi = iplCreateROI(0, moveRect.left, moveRect.top, moveRect.Width(), moveRect.Height());
        if (!pRoi) {
            Stop();
            AfxMessageBox("Couldn't create region-of-interest for motion detection.");
            m_readyFrameIn = FALSE;
            m_frameBusy = FALSE;
            return;
        }

        // Convert the full-color image to an 8-bit/pixel luminance-only image,
        // blur a bit first to lower the noise caused by pixel noise and camera vibration.
        //ZZZ the docs say iplFixedFilter supports in-place, but the routine errored when I tried
to do that.

        iplColorToGray(m_pIplIn, m_pIplTmp);
        iplFixedFilter(m_pIplTmp, m_pIplMotion, IPL_GAUSSIAN_3x3);

        //ZZZ I want to add an edge-enhancement to avoid triggering on shadows or gross lighting c
hanges,
        //ZZZ but don't have time at the moment to build it.

        // If we're just starting,
        // copy this frame to the reference so we don't falsely trigger on motion.

        if (!m_startFrameSeen) {
            iplCopy(m_pIplMotion, m_pIplMotionRef);
        }

        // Find how different the current image is from the motion-detection reference.
        // pixel_is_different = abs(a - b) > (pixel_noise + pixel_noise).
        // This function essentially says "how many corresponding pixels differ by greater than th
e noise in each pixel?"
        //
        // Since iplSubtract() performs saturation arithmetic,
```

```
        // we have to perform two subtractions and sum their results
        // in order to implement our function.

        // Limit the region of interest to our motion-detection rectangle,
        // and limit the division to its number of pixels.
        //ZZZ can we just use iplSetRoi() on the images' ROI's instead of the pointer-setting stuf
f?
        //ZZZ the documentation really isn't explicit about how to set the ROI of an image.

        prevRoi1 = m_pIplMotion->roi;
        m_pIplMotion->roi = pRoi;
        prevRoi2 = m_pIplMotionRef->roi;
        m_pIplMotionRef->roi = pRoi;
        prevRoi3 = m_pIplTmp->roi;
        m_pIplTmp->roi = pRoi;

        iplSubtract(m_pIplMotion, m_pIplMotionRef, m_pIplTmp);
        iplThreshold(m_pIplTmp, m_pIplTmp, diffThreshold);
        fDiff = iplNorm(m_pIplTmp, NULL, IPL_L1) / 255.0;

        iplSubtract(m_pIplMotionRef, m_pIplMotion, m_pIplTmp);
        iplThreshold(m_pIplTmp, m_pIplTmp, diffThreshold);
        fDiff += iplNorm(m_pIplTmp, NULL, IPL_L1) / 255.0;

        // Restore the regions of interest.

        m_pIplMotion->roi = prevRoi1;
        m_pIplMotionRef->roi = prevRoi2;
        m_pIplTmp->roi = prevRoi3;

        // fDiff is now the number of pixels that differed.
        // Convert it to the percent of pixels that differed.

        fDiff /= ((double) moveRect.Width() * (double) moveRect.Height());
        m_fCurrentMotion = fDiff * 100.0;

        // If enough pixels have changed, call it motion.
        // Note when our most recent motion occurred
        // and that our change-detection algorithm doesn't have to wait for motion to occur any mo
re.

        if (m_fCurrentMotion > m_fMotionThreshold) {
            motionOccurred = TRUE;
            m_msPrevMoveTime = startTime;
            m_waitingForMovement = FALSE;
        }

        // Now that we're done with the previous edge-enhanced motion reference,
        // update it.

        iplCopy(m_pIplMotion, m_pIplMotionRef);

        switch (m_snapMode) {
        case SNAP_BULB:
        case SNAP_TIME_LAPSE:
        case SNAP_ON_MOTION:
            // these are not change-relative cases.
            break;

        case SNAP_ON_NOVELTY:
        case SNAP_ON_CHANGE:
```

```
        // Find the number of pixels that have changed
        // from our previous change-reference to our current frame,
        // using the same algorithm as above.

        // If we're just starting,
        // copy this frame to the reference so we don't falsely trigger on change.

        if (!m_startFrameSeen) {
            iplCopy(m_pIplMotion, m_pIplChangeRef);
        }

        // Limit the region of interest to our motion-detection rectangle,
        // and limit the division to its number of pixels.

        prevRoi1 = m_pIplMotion->roi;
        m_pIplMotion->roi = pRoi;
        prevRoi2 = m_pIplChangeRef->roi;
        m_pIplChangeRef->roi = pRoi;
        prevRoi3 = m_pIplTmp->roi;
        m_pIplTmp->roi = pRoi;

        iplSubtract(m_pIplMotion, m_pIplChangeRef, m_pIplTmp);
        iplThreshold(m_pIplTmp, m_pIplTmp, diffThreshold);
        fDiff = iplNorm(m_pIplTmp, NULL, IPL_L1) / 255.0;

        iplSubtract(m_pIplChangeRef, m_pIplMotion, m_pIplTmp);
        iplThreshold(m_pIplTmp, m_pIplTmp, diffThreshold);
        fDiff += iplNorm(m_pIplTmp, NULL, IPL_L1) / 255.0;

        // Restore the regions of interest.

        m_pIplMotion->roi = prevRoi1;
        m_pIplChangeRef->roi = prevRoi2;
        m_pIplTmp->roi = prevRoi3;

        // fDiff is now the number of pixels that differed.
        // Convert it to the percent of pixels that differed, expressed in 10ths of a percent.

        fDiff /= ((double) moveRect.Width() * (double) moveRect.Height());
        pcNumChanged = fDiff * 100.0;

        // If enough pixels have changed,
        // this is novelty (change from the last stable change).

        if (pcNumChanged > m_fMotionThreshold) {
            noveltyOccurred = TRUE;
        }

        // See how long it's been since motion happened.
        // If it's been long enough, and we aren't waiting for some motion to happen first,
        // we can see if a stable change has happened.

        msIdle = startTime - m_msPrevMoveTime;
        if ((!m_waitingForMovement && msIdle > m_msIdleThreshold) || m_forceSnap) {
            m_waitingForMovement = TRUE;     // (because starting with the next frame, we're wa
iting for new movement)

            // If enough pixels have changed from the last stable scene, call it a stable chan
ge.
            // and record it as the reference for detecting future changes.

            if ((pcNumChanged > m_fMotionThreshold) || m_forceSnap) {
```

```
                changeOccurred = TRUE;
                iplCopy(m_pIplMotion, m_pIplChangeRef);
            }
        }

        break;
    default:    // unknown snap mode.
        Stop();
        ASSERT(FALSE);
    }

    // Now we're done with the region of interest for motion-detection.

    if (!pRoi) {
        Stop();
        ASSERT(FALSE);
    }
    iplDeleteROI(pRoi);
    pRoi = NULL;

    break;
default:    // unknown snap mode.
    Stop();
    ASSERT(FALSE);
}

// Now that we know whether or not there's motion and change,
// decide whether we need to send the image.

switch (m_snapMode) {
case SNAP_BULB:
case SNAP_TIME_LAPSE:
    // These are non-motion modes, handled elsewhere.
    break;
case SNAP_ON_MOTION:
    // If motion happened, send the frame.
    if (motionOccurred) {
        sendFrame = TRUE;
    }
    break;
case SNAP_ON_NOVELTY:
    // If motion occurred relative to the last stable change,
    // send the frame.

    if (noveltyOccurred) {
        sendFrame = TRUE;
    }
    break;
case SNAP_ON_CHANGE:
    // If the image is stable and different enough from the last stable change,
    // send the frame.

    if (changeOccurred) {
        sendFrame = TRUE;
    }
    break;
default:    // unknown snap mode.
    Stop();
    ASSERT(FALSE);
}

// If the user has asked for image histogram equalization, do it.
```

```
// We do this AFTER all the motion-detection
// because histogram equalization increases the pixel noise as the image gets more uniform,
// ultimately converting a black image into full-brightness-range noise.

if (m_doHistogramEQ) {
    // intialize the lookup tables for each channel.

    //ASSERT(m_pIplIn->numchannels == 3);
    for (lutIdx = 0; lutIdx < 3; ++lutIdx) {
        pLut[lutIdx] = &lut[lutIdx];

        lut[lutIdx].num = 256;
        lut[lutIdx].key = &lutKey[256 * lutIdx];
        lut[lutIdx].value = &lutValue[256 * lutIdx];
        lut[lutIdx].factor = NULL;
        lut[lutIdx].interpolateType = IPL_LUT_INTER;

        for (keyIdx = 0; keyIdx < 256; ++keyIdx) {
            lutKey[lutIdx * 256 + keyIdx] = keyIdx;
            lutValue[lutIdx * 256 + keyIdx] = 0;
        }
    }

    // Calculate the image histogram,
    // then equalize that histogram and apply it to the image.

    //ZZZ the pre-equalized histogram could be useful for detecting that, for example,
    //ZZZ the image is totally black and isn't worth sending.

    iplComputeHisto(m_pIplIn, &pLut[0]);
    iplHistoEqualize(m_pIplIn, m_pIplIn, &pLut[0]);
}

// If this frame is to be forced, send it.

if (m_forceSnap) {
    sendFrame = TRUE;
}

// If we've decided to send this frame,
// create a fresh IPL copy of it and hand that copy off to the desired window.

if (sendFrame) {
    IplImage *pIpl = NULL;      // the IPL image copy to send away.

    pIpl = iplCloneImage (m_pIplIn);
    if (!pIpl) {
        Stop();
        AfxMessageBox("Couldn't create a duplicate of the captured image");
        m_readyFrameIn = FALSE;
        m_frameBusy = FALSE;
        return;
    }

    // Tag the copy with whatever text the sender wanted.
    // We don't tag the original because the tag would only flicker past in the preview.

    cStrTag = m_labelText;

    if (m_doLabelTime) {
        if (!cStrTag.IsEmpty()) {
            cStrTag += " ";
```

```
            }
            CString t = ctNow.Format("%I:%M%p"); // (hr:min am/pm)
            t.MakeLower();
            cStrTag += t;
        }

        if (m_doLabelDate) {
            if (!cStrTag.IsEmpty()) {
                cStrTag += " ";
            }
            cStrTag += ctNow.Format("%B %d");    // (%B = full month name, %b = abbreviated month n
ame, %d = day of month)
        }

        // If there's nothing to tag, don't tag the image.

        if (!cStrTag.IsEmpty()) {
            pMsg = TagIplImage(pIpl, (LPCSTR) cStrTag, m_labelColorBk, m_labelColorText);
            if (pMsg) {
                Stop();
                AfxMessageBox(pMsg);
                iplDeallocate(pIpl,IPL_IMAGE_HEADER | IPL_IMAGE_DATA);
                pIpl = NULL;
                m_readyFrameIn = FALSE;
                m_frameBusy = FALSE;
                return;
            }
        }

        // Make this frame "pending".
        // If we can't send it right away, we'll send it when we can.
        // This arrangement makes motion- and change-detection modes be properly throttled by m_ms
TimeLapse:
        // If motion or change occurred during the time-lapse interval,
        // that moved or changed frame is sent once the interval expires.

        if (m_pIplPending) {
            iplDeallocate(m_pIplPending,IPL_IMAGE_HEADER | IPL_IMAGE_DATA);
            m_pIplPending = NULL;
        }
        m_pIplPending = pIpl;
        pIpl = NULL;     // (since we've moved it to Pending)
    }

    // If the time-lapse has expired since we last sent a frame (or the user forced this frame),
    // and we have a frame to send,
    // send or write that pending frame and note the time we sent/wrote it.

    if (m_forceSnap) {
        if (!m_pIplPending) {    // (forcing a frame had better force the frame into pending state)
            Stop();
            ASSERT(FALSE);
        }
    }
    if ((m_forceSnap || msTimeSincePrev >= m_msTimeLapse) && m_pIplPending) {
        m_msPrevSnapTime = startTime;

        if (!m_saveToName.IsEmpty()) {
            pMsg = WriteIpl(m_pIplPending, m_saveToName, m_jpegQuality);
            if (pMsg) {
                Stop();
                AfxMessageBox(pMsg);
```

```
                iplDeallocate(m_pIplPending,IPL_IMAGE_HEADER | IPL_IMAGE_DATA);
                m_pIplPending = NULL;
                m_readyFrameIn = FALSE;
                m_frameBusy = FALSE;
                return;
            }
        }
        //m_pWndNotify->PostMessage(m_msgNotify, 0, (LPARAM) m_pIplPending); // (a SendMessage() w
ould confuse the threads)
        iplDeallocate(m_pIplPending,IPL_IMAGE_HEADER | IPL_IMAGE_DATA);
        m_pIplPending = NULL;

        // Tell our container that an image file is ready for them.
        FireImageReady();
    }

    // Now we can forget whether the user forced this frame
    // and whether this was the first frame after a Start().

    m_forceSnap = FALSE;
    m_startFrameSeen = TRUE;

    // Note how long it took to process this frame.
    endTime = timeGetTime();
    m_frameTime = endTime - startTime;

    // let our preview window know it's time to redraw.
    InvalidateControl();

    m_readyFrameIn = FALSE; // (because we're done processing the current frame)

    m_frameBusy = FALSE;
}
```

Attorney's Docket No.: 042390.P7389 PATENT

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION
(FOR **INTEL CORPORATION** PATENT APPLICATIONS)

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below, next to my name.

I believe I am the original, first, and sole inventor (if only one name is listed below) or an original, first, and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

A MOTION DETECTING WEB CAMERA SYSTEM

the specification of which

__XX__ is attached hereto.
_____ was filed on _____ as
United States Application Number _____
or PCT International Application Number_____
and was amended on _____.
(if applicable)

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claim(s), as amended by any amendment referred to above. I do not know and do not believe that the claimed invention was ever known or used in the United States of America before my invention thereof, or patented or described in any printed publication in any country before my invention thereof or more than one year prior to this application, that the same was not in public use or on sale in the United States of America more than one year prior to this application, and that the invention has not been patented or made the subject of an inventor's certificate issued before the date of this application in any country foreign to the United States of America on an application filed by me or my legal representatives or assigns more than twelve months (for a utility patent application) or six months (for a design patent application) prior to this application.

I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119(a)-(d), of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)			Priority Claimed	
(Number)	(Country)	(Day/Month/Year Filed)	Yes	No
(Number)	(Country)	(Day/Month/Year Filed)	Yes	No

I hereby claim the benefit under Title 35, United States Code, Section 119(e) of any United States provisional application(s) listed below:

Application Number	Filing Date

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

Application Number	Filing Date	Status -- patented, pending, abandoned

I hereby appoint the persons listed on Appendix A hereto (which is incorporated by reference and a part of this document) as my respective patent attorneys and patent agents, with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith.

Send correspondence to Bradley J. Bereznak, BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP, 12400 Wilshire Boulevard 7th Floor, Los Angeles, California 90025 and direct telephone calls to Bradley J. Bereznak, (408) 720-8598.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor Bradford H. Needham

Inventor's Signature _____ Date September 10, 1999

Residence North Plains, Oregon (City, State) Citizenship U.S.A. (Country)

Post Office Address 26955 NW Dorland Road
North Plains, OR 97133

APPENDIX A

William E. Alford, Reg. No. 37,764; Farzad E. Amini, Reg. No. P42,261; Aloysius T. C. AuYeung, Reg. No. 35,432; William Thomas Babbitt, Reg. No. 39,591; Carol F. Barry, Reg. No. 41,600; Jordan Michael Becker, Reg. No. 39,602; Bradley J. Bereznak, Reg. No. 33,474; Michael A. Bernadicou, Reg. No. 35,934; Roger W. Blakely, Jr., Reg. No. 25,831; Gregory D. Caldwell, Reg. No. 39,926; Ronald C. Card, Reg. No. P44,587; Yong S. Choi, Reg. No. P43,324; Thomas M. Coester, Reg. No. 39,637; Michael Anthony DeSanctis, Reg. No. 39,957; Daniel M. De Vos, Reg. No. 37,813; Robert Andrew Diehl, Reg. No. 40,992; Tarek N. Fahmi, Reg. No. 41,402; James Y. Go, Reg. No. 40,621; Willmore F. Holbrow III, Reg. No. P41,845; George W Hoover II, Reg. No. 32,992; Eric S. Hyman, Reg. No. 30,139; Dag H. Johansen, Reg. No. 36,172; William W. Kidd, Reg. No. 31,772; Michael J. Mallie, Reg. No. 36,591; Andre L. Marais, under 37 C.F.R. § 10.9(b); Paul A. Mendonsa, Reg. No. 42,879; Darren J. Milliken, Reg. 42,004; Lisa A. Norris, Reg. No. P44,976; Thien T. Nguyen, Reg. No. 43,835; Thinh V. Nguyen, Reg. No. 42,034; Dennis A. Nicholls, Reg. No. 42,036; Kimberley G. Nobles, Reg. No. 38,255; Daniel E. Ovanezian, Reg. No. 41,236; Babak Redjaian, Reg. No. 42,096; James H. Salter, Reg. No. 35,668; William W. Schaal, Reg. No. 39,018; James C. Scheller, Reg. No. 31,195; Anand Sethuraman, Reg. No. P43,351; Charles E. Shemwell, Reg. No. 40,171; Jeffrey Sam Smith, Reg. No. 39,377; Maria McCormack Sobrino, Reg. No. 31,639; Stanley W. Sokoloff, Reg. No. 25,128; Judith A. Szepesi, Reg. No. 39,393; Vincent P. Tassinari, Reg. No. 42,179; Edwin H. Taylor, Reg. No. 25,129; George G. C. Tseng, Reg. No. 41,355; Joseph A. Twarowski, Reg. No. 42,191; Lester J. Vincent, Reg. No. 31,460; Glenn E. Von Tersch, Reg. No. 41,364; John Patrick Ward, Reg. No. 40,216; Charles T. J. Weigell, Reg. No. 43,398; Kirk D. Williams, Reg. No. 42,229; James M. Wu, Reg. No. P45,241; Steven D. Yates, Reg. No. 42,242; Ben J. Yorks, Reg. No. 33,609; and Norman Zafman, Reg. No. 26,250; my patent attorneys, and James A. Henry, Reg. No. 41,064; my patent agent, of BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP, with offices located at 12400 Wilshire Boulevard, 7th Floor, Los Angeles, California 90025, telephone (310) 207-3800, and Alan K. Aldous, Reg. No. 31,905; Robert D. Anderson, Reg. No. 33,826; Joseph R. Bond, Reg. No. 36,458; Richard C. Calderwood, Reg. No. 35,468; Jeffrey S. Draeger, Reg. No. 41,000; Cynthia Thomas Faatz, Reg No. 39,973; Sean Fitzgerald, Reg. No. 32,027; Seth Z. Kalson, Reg. No. 40,670; David J. Kaplan, Reg. No. 41,105; Charles A. Mirho, Reg. No. 41,199; Leo V. Novakoski, Reg. No. 37,198; Naomi Obinata, Reg. No. 39,320; Thomas C. Reynolds, Reg. No. 32,488; Mark Seeley, Reg. No. 32,299; Steven P. Skabrat, Reg. No. 36,279; Howard A. Skaist, Reg. No. 36,008; Steven C. Stewart, Reg. No. 33,555; Raymond J. Werner, Reg. No. 34,752; and Charles K. Young, Reg. No. 39,435; my patent attorneys, and Thomas Raleigh Lane, Reg. No. 42,781; Calvin E. Wells, Reg. No. P43,256; and Peter Lam, Reg. No. P44,855; my patent agents, of INTEL CORPORATION; and James R. Thein, Reg. No. 31,710, my patent attorney; with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith.

APPENDIX B

Title 37, Code of Federal Regulations, Section 1.56
Duty to Disclose Information Material to Patentability

(a) A patent by its very nature is affected with a public interest. The public interest is best served, and the most effective patent examination occurs when, at the time an application is being examined, the Office is aware of and evaluates the teachings of all information material to patentability. Each individual associated with the filing and prosecution of a patent application has a duty of candor and good faith in dealing with the Office, which includes a duty to disclose to the Office all information known to that individual to be material to patentability as defined in this section. The duty to disclosure information exists with respect to each pending claim until the claim is cancelled or withdrawn from consideration, or the application becomes abandoned. Information material to the patentability of a claim that is cancelled or withdrawn from consideration need not be submitted if the information is not material to the patentability of any claim remaining under consideration in the application. There is no duty to submit information which is not material to the patentability of any existing claim. The duty to disclosure all information known to be material to patentability is deemed to be satisfied if all information known to be material to patentability of any claim issued in a patent was cited by the Office or submitted to the Office in the manner prescribed by §§1.97(b)-(d) and 1.98. However, no patent will be granted on an application in connection with which fraud on the Office was practiced or attempted or the duty of disclosure was violated through bad faith or intentional misconduct. The Office encourages applicants to carefully examine:

(1) Prior art cited in search reports of a foreign patent office in a counterpart application, and

(2) The closest information over which individuals associated with the filing or prosecution of a patent application believe any pending claim patentably defines, to make sure that any material information contained therein is disclosed to the Office.

(b) Under this section, information is material to patentability when it is not cumulative to information already of record or being made or record in the application, and

(1) It establishes, by itself or in combination with other information, a prima facie case of unpatentability of a claim; or

(2) It refutes, or is inconsistent with, a position the applicant takes in:

(i) Opposing an argument of unpatentability relied on by the Office, or

(ii) Asserting an argument of patentability.

A prima facie case of unpatentability is established when the information compels a conclusion that a claim is unpatentable under the preponderance of evidence, burden-of-proof standard, giving each term in the claim its broadest reasonable construction consistent with the specification, and before any consideration is given to evidence which may be submitted in an attempt to establish a contrary conclusion of patentability.

(c) Individuals associated with the filing or prosecution of a patent application within the meaning of this section are:

(1) Each inventor named in the application;

(2) Each attorney or agent who prepares or prosecutes the application; and

(3) Every other person who is substantively involved in the preparation or prosecution of the application and who is associated with the inventor, with the assignee or with anyone to whom there is an obligation to assign the application.

(d) Individuals other than the attorney, agent or inventor may comply with this section by disclosing information to the attorney, agent, or inventor.